ACM SIGACT News Distributed Computing Column 9

Sergio Rajsbaum*

October 31, 2002

Abstract

The Distributed Computing Column covers the theory of systems that are composed of a number of interacting computing elements. These include problems of communication and networking, databases, distributed shared memory, multiprocessor architectures, operating systems, verification, internet, and the web.

This issue consists of the paper "Incentives and Internet Computation" by Joan Feigenbaum and Scott Shenker. Many thanks to them for contributing to this issue.

Request for Collaborations: Please send me any suggestions for material I should be including in this column, including news and communications, open problems, and authors willing to write a guest column or to review an event related to theory of distributed computing.

Incentives and Internet Computation

Joan Feigenbaum
Computer Science Department
Yale University
New Haven, CT 06520 USA
feigenbaum@cs.yale.edu

Scott Shenker
ICSI
1947 Center Street
Berkeley, CA 94704 USA
shenker@icsi.berkeley.edu

1 Introduction

The emergence of the Internet as a standard platform for distributed computing has led to diversification of the research agenda in distributed algorithms, and that agenda now consists of much more than traditional, core PODC concerns. If distributed algorithms are to be designed, analyzed, implemented, and deployed for the full range of applications that are now plausible, the research community will need to develop new computational models, new failure models, new measures of computational complexity, and new analysis techniques. This column is intended as an introduction to one theme that has grown steadily in popularity and importance during the past few years: the recognition that participants in an Internet algorithm are economic actors as well as computational processes.

*Instituto de Matemáticas, UNAM. Ciudad Universitaria, Mexico City, D.F. 04510 rajsbaum@math.unam.mx.

Multi-agent systems have been extensively studied in both economics and computer science, but the two communities have approached the topic very differently. In traditional theoretical computer science (TCS), computational agents are typically assumed either to be *obedient* (*i.e.*, to follow the prescribed algorithm) or to be *adversaries* who "play against" each other. On the other hand, the *strategic* agents in game theory are neither obedient nor adversarial. Although one cannot assume that they will follow the prescribed algorithm, one can assume that they will respond to incentives. Thus, the economics literature traditionally stressed incentives and downplayed computational complexity, and the TCS literature traditionally did the opposite. The opportunity to design, implement, and deploy algorithms on a widely used distributed-computing platform provides strong motivation to develop a unified approach to incentive compatibility and computational tractability: Ownership, operation, and use by many self-interested, independent parties give the Internet the characteristics of an economy as well as those of a computer.

Although many subdisciplines of computer science have a long history of using game theory — such as networking (*e.g.*, [11, 18]), distributed artificial intelligence (*e.g.*, [24, 26]), and market-based computation (*e.g.*, [29]) — the first work in TCS that explicitly addressed incentives and computational complexity simultaneously was Nisan and Ronen's seminal paper [23] on *algorithmic mechanism design* (AMD). That paper put forth a formal model of *centralized* computation that combined incentive compatibility (the "mechanism design" part) with computational tractability (the "algorithmic" part). Feigenbaum, Papadimitriou, and Shenker [8] extended this to *distributed algorithmic mechanism design* (DAMD), in which the same goals of incentive compatibility and computational tractability are present, but, in addition, the agents, the relevant information, and the computational model are all inherently distributed.

The Internet is an arena in which incentive compatibility, distributed computation, and computational complexity are all highly relevant. Thus, we believe that DAMD, with its simultaneous attention to these issues, will be important for understanding our Internet-centric future. This column provides a basic overview of DAMD and identifies several promising areas for future research. We start, in Section 2, by providing some necessary background on *mechanism design* (MD), algorithmic and otherwise. In Sections 3 and 4, we review previous DAMD results on multicast cost sharing and interdomain routing, respectively. In Section 5, we present some general open questions about the meaning of "hardness" and "easiness" of Internet computation. In Section 6, we present our case for the importance to computer science of *indirect* mechanisms, which are *not* a central part of the Economics research agenda in AMD. For a more in-depth survey of the AMD research agenda, please see [9] and the papers referred to therein.

2 MD to AMD to DAMD

In essence, game theory is the study of what happens when independent agents act selfishly. Mechanism design asks how one can design systems so that agents' selfish behavior results in the desired system-wide goals. The "mechanisms" in this field are output specifications and payments to agents that incentivize them to behave in ways that lead to the desired system-wide result. For example, consider the problem of routing. Agents may be individual routers within a network or entire autonomous *domains*. Each agent incurs a cost when it transports a packet, and this cost is known only to the agent, not to the mechanism designer or to the routing protocol. Each agent is required by the protocol to declare a cost. The system-wide goal is to have the routing protocol choose the true lowest-cost path between any two agents in the network. The mechanism specifies, for each network topology, each sender-receiver pair, and each set of agents' declared costs, a path from sender to receiver and a payment to each agent; the mechanism designer's task is to find a formula for the payments that causes agents to be no worse off by revealing their true costs than they would be by lying about their costs. Such truthful revelation would allow the routing protocol to achieve the system-wide goal of having all the traffic follow lowest-cost paths.

More formally, consider a distributed system in which there is a set of possible outcomes $\mathcal{O}$. Each of the n autonomous strategic agents has a utility function $u_i : \mathcal{O} \to \Re$, where $u_i \in \mathcal{U}$, that expresses its preferences over these outcomes. The desired system-wide goals are specified by a *social choice function* (SCF) $F : \mathcal{U}^n \to \mathcal{O}$ that maps each particular instantiation of agents (who are completely described by their utility functions) into a particular outcome.[1] The problem is that these utilities are known only to the agents, not to the system designer or to any other central administrative entity; thus, one cannot just implement the desired outcome by fiat.

An SCF is *strategyproof* if $u_i(F(u)) \geq u_i(F(u|^i v))$, for all i and all $v \in \mathcal{U}$, where we use the notation $(u|^i v)_i = v$ and $(u|^i v)_j = u_j$, for all $j \neq i$. If F is strategyproof, then no agent has an incentive to lie, and the desired social goals can be achieved by asking agents to reveal their utility functions. Mechanisms in which agents are asked to directly reveal their utility functions are call *direct* mechanisms.

An SCF is *group-strategyproof* if the following holds for all S, u, and u' (where $S = \{i \mid u_i \neq u'_i\}$ is the defecting group): Either $u_i(F(u)) = u_i(F(u'))$, $\forall i \in S$, or $\exists i \in S$ for which $u_i(F(u')) < u_i(F(u))$. That is, if any agent in the group benefits from the group's colluding and lying to the mechanism, then at least one agent in the group suffers.

An important class of problems are those in which the utilities are *quasilinear*, and the outcome space $\mathcal{O}$ factors into a set of system states $\tilde{\mathcal{O}}$ and a set of payment states $\mathcal{P} \subseteq \Re^n$ that represent a vector of payoffs (or charges). At a particular outcome $o = (\tilde{o}, p)$, agent i's utility factors into $u_i(o) = v_i(\tilde{o}) + p_i$, where $v_i : \tilde{\mathcal{O}} \to \Re$ represents his valuations of each of the system states, and p_i is his payment. For such problems, there is a class of strategyproof mechanisms, called Vickrey-Clarke-Groves (VCG) mechanisms [3, 15, 28], that result in the system state that optimizes $\sum_i v_i(\tilde{o})$.

Direct strategyproof mechanisms provide a conceptually simple, if not always ideal (see Section 6), way to achieve strategyproof SCFs. However, there are many cases in which the desired result, *i.e.*, the desired social choice function F, is not strategyproof. To describe how to realize such nonstrategyproof SCFs, we now introduce *indirect* mechanisms. Here, one designs a mechanism $< M, S >$, where S is a *strategy space*, and $M : S^n \to \mathcal{O}$ maps vectors of strategies into outcomes.[2] These are called indirect mechanisms, because the agents no longer directly reveal their utilities but instead choose strategies from the space S. This strategy choice is done selfishly, with each agent attempting to maximize its own utility. For a given mechanism M and a given utility vector u, we let the set $C_M(u) \subseteq S^n$ represent all possible strategy vectors that could reasonably result from selfish behavior. This set is called the *solution concept*. Traditional game theory often uses the *Nash-equilibrium* solution concept, *i.e.*, selfish play is assumed to result in strategy vectors in which no agent can unilaterally increase his utility. Other solution concepts include *rationalizable strategies* (agents use strategies that are best responses to rational beliefs about the other agents' strategy choices [2, 25]), *evolutionarily stable strategies* (agents imitate the successful strategies used by others in previous rounds of the game [27]), and *dominant strategies* (agents only choose strategies that, *regardless* of how other agents play, never result in lower payoffs than any other strategy). To date, most of the AMD and DAMD literature uses the dominant-strategy solution

[1] More generally, we can consider *social choice correspondences* (SCCs), $H : \mathcal{U}^n \to 2^{\mathcal{O}}$, which map utility vectors into sets of outcomes. For notational simplicity, we discuss only SCFs in this section. In addition, we restrict ourselves to equivalent agents; in general, each agent could have a different set of possible utilities $\mathcal{U}_i$.

[2] Our assumption that all agents are equivalent, made for notational simplicity, renders all strategy spaces the same; in general, we could have different strategy spaces S_i.

concept.

The goal of mechanism design is to define a mechanism M that implements the SCF, *i.e.*, $M(C_M(u)) = F(u)$, for all $u \in \mathcal{U}^n$.

When this condition holds, then selfish behavior by the agents will result in the desired system-wide outcome. In short, the system will be incentive-compatible. There is a large game-theory literature on which SCFs can be achieved for different notions of "incentive compatibility," *e.g.*, for different solution concepts; see Jackson [17] for an overview. With the Nash-equilibrium solution concept, one can design mechanisms to achieve a very wide range of non-strategyproof social choice functions [19].

When $M = F$ and $S = \mathcal{U}$, we reduce to the direct-mechanism case, and so our preceding discussion applies to direct mechanisms as well. That is, one can achieve nonstrategyproof SCFs with direct mechanisms by invoking different solution concepts. For example, one can achieve efficiency and budget balance using the Bayesian-Nash-equilibrium solution concept [1, 4] – something that is impossible using the dominant-strategy solution concept [13].

It is important to note that, although the mechanism is chosen by the system designer, the solution concept is supposed to reflect reality. The solution concept thus depends greatly on the context (*e.g.*, is it a repeated game or a single-shot game, do agents collude, do they know about the other agents, do they know about the other agents' strategic choices, *etc.*). Because the Internet is somewhat different from traditional game-theoretic contexts, the traditional solution concepts may not be sufficient; this issue is discussed in Section 8 of [9].

The game-theory literature on mechanism design does not consider computational and communication complexity, and many of the existence proofs rely on extremely impractical mechanisms. For the mechanism-design approach to have any practical relevance for Internet computation, one must focus on scalable algorithms. That is, the function M must be computable with reasonable computational and communication resources.

Nisan and Ronen [23] initiated the study of AMD by adding computational tractability to the set of concerns that must be addressed in the design of incentive-compatible mechanisms. Succinctly stated, Nisan and Ronen's contribution to the mechanism-design framework is the notion of a (centralized) *polynomial-time mechanism*, *i.e.*, one in which $M(\cdot)$ is polynomial-time computable. They also provide strategyproof, polynomial-time VCG mechanisms for some concrete problems, including lowest-cost paths and task allocation.

The centralized computational model of [23] is not adequate for the study of Internet computation, where not only are the agents distributed, but so are the resources (*e.g.*, link bandwidth and cache storage) and the computational nodes. Internet-based mechanisms involve distributed algorithms and any measure of their computational feasibility must reflect their distributed nature. In one attempt to address this issue, Feigenbaum, Papadimitriou, and Shenker [8] put forth a general concept of network complexity that requires a distributed algorithm executed over an interconnection network T to be modest in four respects: the total number of messages sent over T (ideally, this should be linear in $|T|$), the maximum number of messages sent over any one link in T (ideally, this should be constant, to avoid "hot spots" altogether), the maximum size of a message, and the local computational burden on agents.

The network-complexity criterion in [8] evaluates the mechanism in isolation based on its *absolute* computation and communication requirements. A *relative* notion of complexity, which we call *protocol compatibility*, is adopted in the work of Feigenbaum, Papadimitriou, Sami, and Shenker [7] on interdomain-routing mechanism design. This measure of complexity does not place absolute limits on what is considered feasible; instead, it requires the mechanism to be a simple extension of a widely deployed, standard Internet protocol. The relevant standardized protocol in [7] is the

Border Gateway Protocol (BGP). For a distributed algorithm that computes a mechanism to be considered a simple extension of a standard protocol, it must have the same general algorithmic structure as the standard and must not require substantially more computation, communication, local storage, or any other resource expenditure than the standard, regardless of whether the standard has high or low absolute network complexity. Protocol compatibility addresses two aspects of practical feasibility – computational tractability and deployability[3] – and we expect it to become an increasingly important aspect of DAMD in particular and Internet algorithms in general. In this column, we use the term *network complexity* generically, encompassing the absolute notion of network complexity used in [8], the relative notion of protocol compatibility used in [7], and other related notions of complexity of Internet computation that will arise in the analysis of future distributed algorithmic mechanisms. Clearly, "network complexity" is not (yet) a well defined term, and we return to this point in Section 5 below. We expect the development of more *prima facie* good (and bad) distributed algorithmic mechanisms to lead to a satisfactory formalization.

3 Multicast Cost Sharing

The *multicast cost-sharing mechanism-design* problem involves an agent population P residing at a set of network nodes N that are connected by bidirectional network links L. The multicast flow emanates from a source node $\alpha_s \in N$; given any set of receivers $R \subseteq P$, the transmission flows through a *multicast tree* $T(R) \subseteq L$ rooted at α_s and spanning the nodes at which agents in R reside. It is assumed that there is a *universal* tree $T(P)$ and that, for each subset $R \subseteq P$, the multicast tree $T(R)$ is merely the smallest subtree of $T(P)$ required to reach the elements in R. Each link $l \in L$ has an associated (finite) cost $c(l) \geq 0$ that is known by the nodes on each end, and each agent i assigns a value u_i to receiving the transmission. A cost-sharing mechanism determines which agents receive the multicast transmission and how much each receiver is charged. We let $x_i \geq 0$ denote how much agent i is charged and σ_i denote whether agent i receives the transmission; $\sigma_i = 1$ if the agent receives the multicast transmission, and $\sigma_i = 0$ otherwise. We use u to denote the vector $(u_1, u_2, \ldots, u_{|P|})$, and the vector v to denote the set of values "declared" by the agents to the mechanism; these need not, in general, be the same as their true values. The notation u_{-i} is used for the vector of all values *except* u_i. The mechanism M is then a pair of functions $M(v) = (x(v), \sigma(v))$. The *receiver set* for a given input vector is $R(v) = \{i \mid \sigma_i = 1\}$. An agent's individual welfare is therefore given by the quasilinear form $w_i = u_i\sigma_i - x_i$. The cost of the smallest subtree $T(R)$ of $T(P)$ needed to reach a set of receivers $R = R(v)$ is $c(T(R))$, and the overall welfare, also known as *efficiency* or *net worth*, is $NW(R) = v_R - c(T(R))$, where $v_R = \sum_{i \in R} v_i$ and $c(T(R)) = \sum_{l \in T(R)} c(l)$. The overall welfare measures the total benefit of providing the multicast transmission (the sum of the valuations minus the total transmission cost).

In order to ensure that the values declared by the agents are truthful, the mechanism should be strategyproof. However, there are other goals besides strategyproofness that the mechanism designer must consider in this cost-sharing context. One natural goal is to maximize the net worth; an *efficient* mechanism is one in which $\sigma(v)$ maximizes NW for all v. As shown by Green and Laffont [13], all strategyproof and efficient mechanisms take on the special Vickrey-Clarke-Groves (VCG) form:

Theorem 1 *Consider a mechanism $M(v) = (x(v), \sigma(v))$ such that*

- *$\sigma(v)$ maximizes net worth NW*

[3]In practice, straightforward extensions of existing protocols are easier to deploy than *de novo* designs.

- $x_i(v) = h_i(v_{-i}) - \sum_{j \neq i} v_j \sigma_j(v) + C(\sigma(v))$*, for some set of functions* $h_j(v_{-j})$.

This mechanism is strategyproof. Conversely, any strategyproof and efficient mechanism is of this form.

Sketch of Proof: To show that such a mechanism is strategyproof, focus on a particular agent i. Fix v_{-i}, the reported utilities of all agents except i. Let u_i be agent i's true utility and v_i her reported utility. The welfare of agent i is

$$w_i(v) = u_i\sigma_i(v) - x_i(v) = [u_i\sigma_i(v) + \sum_{j \neq i} v_j\sigma_j(v) - C(\sigma(v))] - h_i(v_{-i}) \tag{1}$$

Let $\mu = \sigma(v)$ be the chosen membership vector. Note that, for any given μ, the expression within brackets [] is identical to the expression for $NW((u_i, v_{-i}), \mu)$, *i.e.*, the net worth if all the agents' true utilities were (u_i, v_{-i}), and the membership function chosen was μ. Also note that the dependence of Equation (1) on v_i is limited to the dependence of μ on v_i.

Consider a function $f(\rho) = NW((u_i, v_{-i}), \rho)$ defined over all possible membership vectors ρ. The mechanism we are given is efficient at all utility profiles, and so $f(\rho)$ is maximized by setting $\rho = \sigma(u_i, v_{-i})$. If i sets $v_i = u_i$, then the membership vector chosen by the mechanism will be $\mu = \sigma(u_i, v_{-i})$. This makes the expression within brackets [] in Equation (1) reach its maximum over all possible membership vectors; a fortiori, it must also be the maximum over all possible choices of v_i. Thus, setting $v_i = u_i$ maximizes Equation (1), and the mechanism is strategyproof.

Now consider a strategyproof and efficient mechanism $(x(v), \sigma(v))$. Define $k_i(v) = \sum_{j \neq i} v_j\sigma_j(v) - C(\sigma(v)) - x_i(v)$. If we can show that k_i does not depend on v_i, then we are done. For the rest of the argument, we hold v_{-i} fixed; for convenience, we will not explicitly note the dependence on v_{-i} and instead denote functions as depending only on v_i. In addition, define a cost function $C(\sigma) = \sum_{l \in T(R)} c(l)$. Our proof proceeds in four steps.

First, because $\sigma(v_i)$ is chosen to maximize net worth, there is some value (perhaps infinite, perhaps negative), call it ν, such that $\sigma_i(v_i) = 1$, for all $v_i > \nu$, and $\sigma_i(v_i) = 0$, for all $v_i < \nu$.

Second, consider the function $x_i(v_i)$. By strategyproofness, this function takes on two values, one when $v_i < \nu$ and another when $v_i > \nu$. Moreover, the difference in these two values must be exactly ν, or else the individual welfare $w_i(v_i)$ above and below ν would differ, violating strategyproofness.

Third, consider the function $l_i(v_i) = \sum_{j \neq i} u_j\sigma_j(v_i) - C(\sigma(v_i))$. This is essentially the net worth minus the contribution from agent i. Because $\sigma(v_i)$ is chosen to maximize net worth, this function takes on one value for $v_i < \nu$ and another when $v_i > \nu$. Again, the differences in these two quantities must be exactly ν, because the net worth at the transition point must be continuous.

Fourth, note that $k_i(v_i) = l_i(v_i) - x_i(v_i)$. Because each term in this function takes on only two values, which differ by ν, and the transition occurs at the same place $v_i = \nu$, the function itself must be constant. $\square$

This theorem describes a class of strategyproof and efficient mechanisms. As shown in [20], there is only one such mechanism that has the following two properties:

NPT No Positive Transfers: $x_i(v) \geq 0$, or, in other words, the mechanism cannot *pay* receivers to receive the transmission.

VP Voluntary Participation: $w_i(v) \geq 0$; this implies that $x_i = 0$ whenever $\sigma_i = 0$ and that agents are always free to not receive the transmission and not be charged (by setting $v_i = 0$).

That mechanism, called the *marginal-cost mechanism* (MC), can be defined as follows. For each v, the mechanism chooses the $\sigma(v)$ (or, equivalently, the receiver set R) to maximize NW. Let W

be the net worth of this efficiency-maximizing R. For each $i \in R$, let W^{-i} be the net worth of the receiver set that the MC mechanism would have computed if i had not participated (*i.e.*, if v_i had been set to 0). Then $W - W^{-i}$ measures the gain in overall welfare that results from i's participation. The cost share that MC assigns to i is $x_i \equiv v_i - (W - W^{-i})$.

Another goal that is important in some contexts is *budget balance*; a mechanism is budget-balanced if $\sum_i x_i(v) = C(\sigma(v))$ for all v. Although one would naturally like to achieve both budget balance and efficiency, a classical result of game theory (also due to Green and Laffont [13]) precludes this possibility.

Theorem 2 *There is no strategyproof, efficient, and budget-balanced mechanism.*

The proof of this result, which we omit here, consists of demonstrating that mechanisms of the VCG form cannot be budget-balanced.

There are many strategyproof and budget-balanced mechanisms; in [20], it is shown that, among these, the Shapley (SH) mechanism minimizes the worst-case efficiency loss. SH assigns cost shares x_i by dividing the cost $c(l)$ of each link l in $T(R)$ equally among all members of $i \in R$ that are downstream of l. The SH receiver set is the largest $R \subseteq P$ such that $u_i \geq x_i$, for all $i \in R$.

Although the SH mechanism combines the desirable economic properties of budget balance and strategyproofness, it is hard to compute. More specifically, it is shown in [6] that SH has inherently high network complexity:

Theorem 3 *Any algorithm, deterministic or randomized, that computes SH must, in the worst case, send $\Omega(|P|)$ bits over linearly many links.*

The proof that SH has bad network complexity uses standard lower-bound techniques from communication complexity, and so we will not give it here.

In constrast, as is shown in [8], the MC mechanism has good network complexity. Because the algorithm used in the proof of this result exemplifies many properties that are desirable in the DAMD context, we give the proof here.

Theorem 4 *MC cost sharing requires exactly two messages per link. There is an algorithm that computes the cost shares by performing one bottom-up traversal of $T(P)$, followed by one top-down traversal, and this algorithm is optimal with respect to number of messages sent.*

Proof: In order to describe the algorithm[4], we need the following notation. Let u^α denote the sum of the valuations of the users located at node α, c^α the cost of the link from α to its parent $p(\alpha)$ in the tree $T(P)$, $Ch(\alpha)$ all the child nodes of α in the tree $T(P)$, $V(P)$ all nodes in the tree $T(P)$, $res(\alpha)$ the set of users at node α, and $T^\alpha(P)$ the union of the subtree rooted at α and the link from α to $p(\alpha)$. With this, we can compute $W^\alpha(u)$, which is the welfare (*i.e.*, sum of valuations minus cost) of $T^\alpha(P)$, as follows:

$$W^\alpha(u) = u^\alpha + \left(\sum_{\beta \in Ch(\alpha) | W^\beta(u) \geq 0} W^\beta(u) \right) - c^\alpha.$$

[4]Note that all of the formulas in this algorithm and proof are stated in terms of the true valuations u instead of the declared valuations v. This choice of notation is made to emphasize the fact that MC is strategyproof, and hence one may assume that $u = v$.

Note that these values can be computed by the bottom-up traversal given in Figure 1 below. Naturally, $\sigma(i) = 1$ (that is, user i is included in the multicast) if $W^{\alpha}(u) \geq 0$ for all nodes α in the path from user i to the root.

Once the $W^{\alpha}(u)$'s have been computed, the values of the $\sigma_i(u)$'s – that is, the bits that indicate whether a user i is a member of the efficient set $R^*(u)$ – can be propagated in a top-down traversal.

The cost share $x_i(u)$ for user $i \in R^*(u)$ does not require a from-scratch recomputation of $W^{-i} = W(u|^i0)$. For each node α and user $i \in R^*(u)$ at α, let $y_i(u)$ be the smallest $W^{\beta}(u)$ of any node β in the path from α to the root. (This minimum welfare value might occur at α.) Then there are two cases:

- If $u_i \leq y_i(u)$, then, without user i, the efficient set is the same as it is with user i. That is, $R^*(u) = R^*(u|^i0)$, and the difference $W(u) - W(u|^i0)$ is u_i. Therefore, user i must pay $x_i(u) \equiv u_i - (W(u) - W(u|^i0)) = 0$.

- If, however, $u_i > y_i(u)$, then dropping user i results in the elimination from $R^*(u)$ of a subtree of total welfare $y_i(u)$, and thus user i must pay exactly $x_i(u) = u_i - y_i(u)$.

To see this, note that dropping user i decreases $W^{\alpha}(u)$ by u_i. If $u_i > y_i(u)$, then there is some lowest (furthest from the source) ancestor α' of α for which dropping user i causes $W^{\alpha'}(u)$ to become negative and $T^{\alpha'}(P)$ to be dropped from $R^*(u)$. Dropping $T^{\alpha'}(P)$ may in turn result in a negative welfare value for some ancestor α'' of α', which would cause $T^{\alpha''}(P)$ to be dropped, and so forth. This "chain reaction" stops after the removal of a minimum-welfare subtree $T^{\beta}(P)$. On the other hand, if $u_i < y_i(u)$, then the tree structure of $R^*(u|^i0)$ is the same as that of $R^*(u)$.

Observe that this propagation of $y_i(u)$ can be combined with the propagation of $\sigma_i(u)$, as shown in Figure 2 below.

In the top-down traversal given in Figure 2, we assume that each node α has the "state" from the (earlier) execution of the bottom-up traversal; this consists of the messages that it received from its children, the message that it sent to its parent, and the values σ_i for users i at α (some of which were erroneously, but temporarily, set to 1 and will be corrected in the top-down traversal). The top-down traversal has to convey enough information to allow nodes to compute cost shares and to correct erroneous σ_i values.

Finally, note that two messages must in fact travel over each link in $T(P)$ if cost shares are to be computed correctly. There are instances in which, for all $\alpha \in V(P)$, cost shares at α depend on valuations at every descendant of α and on valuations and/or link costs between α and the root α_s of $T(P)$. In our model, α can only compute such a share after receiving *some* information from its parent and each of its children (although perhaps not as many bits of information as our algorithm sends). Examples of such instances include those in which $R^*(u) = P$ but setting $u_i = 0$ for any i would cause all users in a subtree rooted at some $\beta \in Ch(\alpha_s)$ not to receive the transmission (because each link joining α_s to one of its children has a very high cost). □

Before turning to our next representative DAMD problem, *i.e.*, interdomain routing, we say a few words about why efficiency and budget balance are natural mechanism-design goals. Efficiency arises naturally as a design goal in the scenario in which the network is owned and operated by society at large, and multicast delivery may be subsidized, *e.g.*, via taxation, if the cost-sharing mechanism runs a deficit; here, the MC mechanism is a natural one to use, because it maximizes the overall welfare of the society as a whole and ensures (because it's strategyproof) that, once the collective choice has been made to charge for multicast delivery in this fashion, no single agent can cheat the group. Budget balance arises naturally as a design goal if the prices charged for multicast delivery must be set by competition among service providers. Competing providers could

Figure 1: Bottom-Up Traversal: Computing Welfare Values

At node $\alpha \in V(P)$
    After receiving a message A^β from each child $\beta \in Ch(\alpha)$
        $W^\alpha \leftarrow u^\alpha + (\sum_{\beta \in Ch(\alpha)} A^\beta) - c^\alpha$
        If $W^\alpha \geq 0$ then
        {
            $\sigma_i \leftarrow 1$ for all $i \in res(\alpha)$
            Send W^α to parent $p(\alpha)$
        }
        Else
        {
            $\sigma_i \leftarrow 0$ for all $i \in res(\alpha)$
            Send 0 to parent $p(\alpha)$
        }

not charge more than their real costs, because they would be undercut, nor could they charge less than their real costs, because they would go out of business. These are the two scenarios considered in the work of Moulin and Shenker [20], which provides the economic foundation for [8] and most of the subsequent work on computational aspects of multicast cost sharing. By contrast, in the scenario in which the multicast delivery is done by a monopoly content owner, profit maximization is the natural mechanism-design goal. Fiat *et al.* [10] provide several novel cost-sharing mechanisms for this scenario.

Finally, we note that, in the problem as we have stated it here, the potential receivers are strategic, but the network (*i.e.*, the universal multicast tree $T(P)$) is obedient. In particular, the network nodes are neither in cahoots with nor conspiring against their resident agents, and the various subnetworks are not competing with each other or with the network as a whole. This is an accurate model of the real-world multicasting scenarios discussed above, in which $T(P)$ is operated by society at large, by a service provider with competitors, or by a monopoly content owner. Even in this simplest possible strategic model, determining the inherent network complexity of natural mechanisms is nontrivial. There may be other multicasting scenarios in which more complex strategic models are needed.

Although the multicast cost-sharing problem has been quite useful in establishing the basic conceptual foundations of DAMD, it is neither realistically formulated nor of pressing importance. Interdomain routing, our next example, is both more realistic and more important.

4 Interdomain Routing

The Internet is comprised of many separate administrative domains or *Autonomous Systems* (ASes). Routing between these domains – *i.e.*, interdomain routing – is currently handled by the Border Gateway Protocol (BGP). There has been much research on routing in general and BGP in particular, but most of it takes a traditional protocol-design approach. Recently, Feigenbaum, Papadimitriou, Sami, and Shenker [7] focused on DAMD issues inherent in interdomain routing.

The basic incentive problem involves transit traffic, *i.e.*, traffic neither originating from nor

Figure 2: Top-Down Traversal: Computing Membership Bits and Cost Shares

Initialize: Root α_s sends W^{α_s} to each of its children.
For each $\alpha \in V(P) - \{\alpha_s\}$
    After receiving message A from parent $p(\alpha)$
        //Case 1: $T^\alpha(P) \cap T(R^*(u)) = \emptyset$.
        //Set σ_i's properly at α and propagate non-membership downward.
        If $\sigma_i = 0$, for all $i \in res(\alpha)$, or $A < 0$ then
        {
            $x_i \leftarrow 0$ and $\sigma_i \leftarrow 0$ for all $i \in res(\alpha)$
            send -1 to β for all $\beta \in Ch(\alpha)$
        }
        //Case 2: $T^\alpha(P) \cap T(R^*(u)) \neq \emptyset$.
        //Compute cost shares and propagate minimum welfare value downward.
        Else
        {
            $A \leftarrow \min(A, W^\alpha)$
            For each $i \in res(\alpha)$
                If $u_i \leq A$, then $x_i \leftarrow 0$, else $x_i \leftarrow u_i - A$
            For each $\beta \in Ch(\alpha)$
                Send A to β
        }

destined to the AS that is currently carrying the packets. For the overall efficiency of the network, packets should travel along shortest or, more generally, lowest-cost paths (LCPs). These optimal paths would typically, in general networks, cut across several ASes. However, carrying transit traffic is a burden that ASes would prefer to avoid. The basic problem is simple: Overall system efficiency is maximized when ASe accept transit traffic, but individual domains are happiest when they carry no transit traffic at all.

In the model used in [7], which is an extension of an earlier (centralized) LCP-mechanism model proposed by Nisan and Ronen [23] and studied further by Hershberger and Suri [16], each AS incurs a per-packet *cost* for carrying traffic, where the cost represents the additional load imposed on the internal AS network by this traffic. Furthermore, the model also assumes that, to compensate for these incurred costs, each AS is paid a *price* for carrying transit traffic. The goal is to maximize network efficiency by routing packets along the LCPs. Standard routing protocols (such as BGP) can compute LCPs given a set of AS costs.[5] However, under many pricing schemes, an AS would be better off lying about its costs; such lying would cause traffic to take non-optimal routes and thereby interfere with overall network efficiency.

To prevent this, one needs the pricing scheme to be strategyproof, so that ASes have no incentive to lie about their costs. The pricing scheme should also have the reasonable property that ASes that carry no transit traffic at all receive no payment. It is shown in [7] that there is only one

[5]BGP does not currently consider general path costs; it simply computes *shortest* AS paths in terms of number of AS hops. However, BGP could be trivially modified so that it computes LCPs; in what follows, we assume that this modification has been made.

strategyproof pricing scheme with this property; it is a member of the VCG family. Moreover, a BGP-compatible distributed algorithm is given that computes these prices. This algorithm requires only minor and straightforward modifications of the BGP computational model given by Griffin and Wilfong [14]. Specifically, the algorithm in [7] requires a small constant-factor increase in both the table sizes and the message sizes of BGP, but it does not require any new messages or any new infrastructural or computational capability; in particular, all messages are still sent between neighbors in the AS graph.[6] Similarly, the local computation done by a node in each stage (*i.e.*, between receiving an updated table from a neighbor and, if necessary, sending an update to each of its neighbors) is the same order of magnitude as the BGP local-computation time.

We include the price-computation algorithm here, because it illustrates the principle of protocol compatibility, which we believe will be an extremely important part of DAMD in particular and Internet algorithms generally.

The network has a set of nodes N, $n = \|N\|$, where each node is an AS. There is a set L of (bidirectional) links between nodes in N. We assume that this network, called the *AS graph*, is biconnected; this is not a severe restriction, because the route-selection problem only arises when a node has multiple potential routes to a destination. For any two nodes $i, j \in N$, T_{ij} is the intensity of traffic (number of packets) originating from i destined for j.

Recall that a node k incurs a transit cost c_k for each transit packet it carries. For simplicity, we assume that this cost is independent of which neighbor k received the packet from and which neighbor k sends the packet to, but our approach could be extended to handle a more general case. We write c for the vector $(c_1, \ldots, c_n)$ of all transit costs and c^{-k} for the vector $(c_1, \ldots, c_{k-1}, c_{k+1}, \ldots c_n)$ of all costs except c_k.

Each node k is given a payment p^k to compensate it for carrying transit traffic. In general, this payment can depend on the costs c, the traffic matrix $[T_{ij}]$, and the network topology. Our only assumption is that nodes that carry no transit traffic whatsoever receive no payment.

Our goal is to send each packet along the LCP, according to the true cost vector c. We assume the presence of a routing protocol like BGP that, given a set of node costs c, routes packets along LCPs. Furthermore, we assume that, if there are two LCPs between a particular source and destination, the routing protocol has an appropriate way to break ties. Let $I_k(c; i, j)$ be the indicator function for the LCP from i to j; *i.e.*, $I_k(c; i, j) = 1$, if node k is an intermediate node on the LCP from i to j, and $I_k(c; i, j) = 0$ otherwise. Note that $I_i(c; i, j) = I_j(c; i, j) = 0$; only the *transit* node costs are counted. The objective function we want to minimize is the total cost $V(c)$ of routing all packets:

$$V(c) = \sum_{i,j \in N} T_{ij} \sum_{k \in N} I_k(c; i, j) c_k$$

Minimizing V is equivalent to minimizing, for every $i, j \in N$, the cost of the path between i and j.

Theorem 5 *When routing picks lowest-cost paths, and the network is biconnected, there is a unique strategyproof pricing mechanism that gives no payment to nodes that carry no transit traffic. The payments to transit nodes are of the form* $p^k = \sum_{i,j \in N} T_{ij} p^k_{ij}$, *where*

$$p^k_{ij} = c_k I_k(c; i, j) + \left[\sum_{r \in N} I_r(c|^k\infty; i, j) c_r - \sum_{r \in N} I_r(c; i, j) c_r \right].$$

The proof of this theorem is a fairly straightforward application of the Green and Laffont [13] characterization of VCG mechanisms and can be found in [7].

[6] The tables in [7] contain both LCPs (as do BGP tables) and costs and prices.

Let $P(c;i,j)$ denote the LCP from i to j for the vector of declared costs c, and let $c(i,j)$ denote the cost of this path. Define $P^{-k}(c;i,j)$ to be the lowest-cost k-avoiding path from i to j. Recall that, if there are multiple LCPs between two nodes, the routing mechanism selects one of them in a loop-free manner. *Loop-free* means that the routes are chosen so that the overall set of LCPs from every other node to j forms a tree. In other words, for each destination j, we assume that the LCPs selected form a tree rooted at j; call this tree $T(j)$.

Consider each destination j separately. The BGP table at i contains the LCP to j:

$$P(c;i,j) \equiv v_s, v_{s-1}, \cdots, v_0 = j,$$

and the cost of this path, $c(i,j)$, where $v_s, v_{s-1}, \cdots, v_0$ are the nodes on the LCP to j and $c(i,j) = \sum_{r=1}^{s} c_{v_r}$.

At the beginning of the computation, all the entries of $p_{ij}^{v_r}$ are set to ∞. Whenever any entry of this price array changes, the array and the path $P(c;i,j)$ are sent to all neighbors of i. As long as the network is static, the entries decrease monotonically as the computation progresses. If the network is dynamic, price and route computation start over whenever there is a change.

Note that each node can infer from the routing tables it receives from its neighbors whether a is its parent, child, or neither in the tree $T(j)$, for each neighbor a. When node i receives an updated price from a neighbor a, it performs the following updates to its internal state.

- If a is i's parent in $T(j)$, then i scans the incoming array and updates its own values if necessary:
$$p_{ij}^{v_r} = \min(p_{ij}^{v_r}, p_{aj}^{v_r}) \quad \forall r \le s-1$$
- If a is a child of i in $T(j)$, i updates its payment values using
$$p_{ij}^{v_r} = \min(p_{ij}^{v_r}, p_{aj}^{v_r} + c_i + c_a) \quad \forall r \le s$$
- If a is neither a parent nor a child, i first scans a's updated path to find the nearest common ancestor v_t. Then i performs the following updates:
$$\begin{aligned} \forall r \le t \;\; p_{ij}^{v_r} &= \min(p_{ij}^{v_r}, p_{aj}^{v_r} + c_a + c(a,j) - c(i,j)) \\ \forall r > t \;\; p_{ij}^{v_r} &= \min(p_{ij}^{v_r}, c_k + c_a + c(a,j) - c(i,j)) \end{aligned}$$

As explained above, the overall algorithm is "BGP-compatible," because it has the same basic structure as the existing BGP. All of the communication among ASes takes place via routing-table exchanges between neighbors in the AS graph, and these exchanges are triggered by table updates. The local-computation step performed by an AS after it receives a table update is a form of dynamic programming, the running time of which is comparable to that of the local computation done by the existing BGP. Let d be the maximum, over all i and j, of the number of hops in the lowest-cost path $P(c;i,j)$ and d' be the maximum, over all i, j, k, of the number of hops in the lowest-cost k-avoiding path $P^{-k}(c;i,j)$. Then we have the following:

Theorem 6 *This algorithm computes the VCG prices correctly, uses routing tables of size $O(nd)$ (i.e., imposes only a constant-factor penalty on the BGP routing-table size), and converges in at most max(d, d') stages (i.e., imposes only a small additive penalty on the worst-case BGP convergence time).*

The intuition behind the proof of Theorem 6 is as follows: The critical information that i needs to compute the correct price p_{ij}^k is the cost of $P(c;i,j)$ and the cost of $P^{-k}(c;i,j)$. (Recall that the cost c_k is distributed with LCPs to k, and so it will be known to i before or at the same time as the cost of $P(c;i,j)$.) After these costs have been discovered, the price p_{ij}^k will not change. The key observation is that, for both $P(c;i,j)$ and $P^{-k}(c;i,j)$, all suffixes of the path are also LCPs or minimum-cost k-avoiding paths; moreover, this is true even at intermediate stages of the computation. This observation is used to prove the correctness of the price-update formulas that appear in the algorithm. Furthermore, it is used to show that the costs along these paths are propagated further in each stage and hence that the total number of stages that they need to reach i is at most the maximum number of hops in any of the paths $P(c;i,j)$ or $P^{-k}(c;i,j)$. A complete proof can be found in the full version of [7].

5 Hard and Easy DAMD Problems

The central mission of TCS is to determine which problems are easy and which are hard in relevant computational models. In the Turing-machine model of centralized computation, the (crude) distinction is between polynomial-time solvable problems and those that are NP-hard. In the PRAM model of parallel computation, it is between those problems that are in NC and those that are P-hard. One of the major goals of this study of DAMD foundations is to develop the tools needed to classify relevant problems as easy or hard "to compute incentive-compatibly on the Internet" and to find more natural examples of both hard and easy DAMD problems.

Informally, a DAMD problem can be considered "easy" if it can be solved in a manner that is both incentive-compatible and computationally tractable. The technical definitions of incentive compatibility and computational tractability will depend on the particular problem under consideration.

The discussion in Section 3 shows that *welfare-maximizing multicast cost sharing* is easy when strategyproofness is the incentive-compatibility requirement, and low absolute network complexity is the computational-tractability requirement. The first open problem is to determine how general this result is. Recall that the MC mechanism is the only strategyproof and efficient mechanism that satisfies NPT and VP. If we remove the NPT and VP requirements, then we have the entire family of VCG mechanisms at our disposal. How many of these have reasonable network complexity?

Open Problem 1 *Fully characterize the set of easy welfare-maximizing multicast cost sharing problems.*

Of course, we are interested in far more than just multicast cost sharing, and one of the central DAMD challenges is the search for additional examples.

Open Problem 2 *Design good distributed algorithmic mechanisms to show that natural problems of interest are easy.*

Four particularly promising application areas for the design of distributed algorithmic mechanisms are web caching, overlay networks, peer-to-peer systems, and distributed task allocation. These are discussed in Section 9 of [9].

While easy problems are a field's "successes," hard problems often lead to a deeper understanding of an approach's fundamental limitations. Thus, we are interested in how to define hardness in the DAMD context. Superficially, a problem is *hard* if it cannot be solved in a manner that satisfies both the incentive-compatibility and the computational-tractability requirements. There will be many problems for which this cannot be done; NP-hard problems, for example, cannot be solved in a computationally tractable manner (unless P=NP), and there are no efficient, strategyproof, and budget-balanced solutions to general cost-sharing problems. However, we are not interested in hardness *per se* but rather in hardness that results from the interplay of incentive compatibility and computational complexity. Thus, a more useful distinction is made by defining a DAMD problem to be *canonically hard* if each of these two requirements can be satisfied individually, but they cannot be satisfied simultaneously. Canonical hard problems will help us understand the fundamental nature of hardness in DAMD, as opposed to hardness that results solely from computational issues or solely from incentive issues.

Budget-balanced multicast cost sharing, under a few natural incentive-compatibility restrictions, is canonically hard. Here, the computational-tractability requirement is low absolute network complexity, as it is for welfare-maximizing multicast cost sharing. The incentive-compatibility conditions include the aforementioned group strategyproofness, NPT, and VP and the following two additional requirements:

- CS Consumer Sovereignty: For given $T(P)$[7] and link costs $c(\cdot)$, there exists some κ such that $\sigma_i(v) = 1$ if $v_i \geq \kappa$; this condition ensures that the network cannot exclude any agent who is willing to pay a sufficiently large amount, regardless of other agents' valuations.

- SYM Symmetry: If i and j are at the same node or are at different nodes separated by a zero-cost path, and $v_i = v_j$, then $x_i = x_j$.

The SH mechanism defined in Section 3 is the natural group-strategyproof, budget-balanced mechanism to consider, for reasons discussed at length by Moulin and Shenker [20], but it is only one of several group-strategyproof, budget-balanced mechanisms in the literature that have properties NPT, VP, CS, and SYM It is shown in [6] that no group-strategyproof, budget-balanced multicast cost-sharing mechanism that satisfies conditions NPT, VP, CS, and SYM can have low absolute network complexity.

Thus, for this problem, one cannot simultaneously meet the computational-tractability requirement and the incentive-compatibility requirement. However, one can meet each requirement independently. The SH mechanism satisfies the incentive-compatibility requirement, and one can easily obtain budget-balanced cost-sharing "mechanisms" with low absolute network complexity if incentive issues are ignored. For instance, in one bottom-up pass of $T(P)$, one can compute $V = \sum_{i \in P} v_i$ and $C = \sum_{l \in L} c(l)$. If $C > V$, no one receives the transmission, and the mechanism does one top-down pass to inform all members of P that this is the outcome; if $C \leq V$, everyone receives the transmission, and the mechanism does one top-down pass to communicate the cost share $(C \cdot v_i)/V$ to agent i, for all $i \in P$.

Group-strategyproof, budget-balanced multicast cost sharing with the additional restrictions of NPT, VP, CS, and SYM is the only canonically hard DAMD problem that has been identified so far. To gain a greater understanding of DAMD, we need many more examples.

Open Problem 3 *Find more DAMD problems that are canonically hard.*

These informal descriptions of what we mean by "easy" and "hard" suffice for the analysis of some examples, but a formal framework is needed if we are to go beyond examples and develop a full-fledged "complexity theory of Internet computation."

[7]For brevity, we often use $T(P)$ to denote four components of a multicast cost-sharing problem instance: the node-set N, the link-set L, the locations of the agents, and the multicast-source location α_s.

Open Problem 4 *Define the computational models and computational resources needed to formalize "network complexity," both absolute and relative, and other relevant measures of DAMD complexity. Develop the appropriate notions of "reduction" to show that certain problems are hard or complete for the relevant complexity classes.*

The preceding discussion considered hardness of DAMD problems. One can also consider hardness of AMD problems by using notions of computational tractability that are appropriate in a centralized computational model. However, we are not aware of a canonically hard AMD problem.[8] All unsuccessful attempts to devise computationally tractable, centralized algorithmic mechanisms that we are aware of fail either because incentive compatibility is unattainable (*e.g.*, budget-balanced and welfare-maximizing cost sharing) or because computational tractability is unattainable (*e.g.*, NP-hard welfare maximization in combinatorial auctions) but not because of the interplay of the two. An interesting open question is whether such canonically hard AMD problems exist.

6 Indirect Mechanisms

Most of the work to date on algorithmic mechanisms in the TCS community has focused on strategyproof, direct mechanisms. The underlying premise of this approach is that agents will voluntarily reveal their private information if it can be proven that lying does them no good in the situation addressed by this particular mechanism-design exercise. We question this premise. Indeed, the TCS community generally questions this premise, which it did not invent but rather inherited from the economics community. Revelation of private information may be in an agent's best interest in the particular game at hand, but it may be unacceptable in the broader context.

For example, in the interdomain-routing mechanism of [7] discussed in Section 4 above, ASes are expected to reveal their internal per-packet transit costs, and conventional economic wisdom would have it that they'd be willing to do so, because the mechanism is strategyproof. However, this seems unrealistic: Revealing its true transit costs may reveal details about an AS's internal network that it wants to keep private for reasons that have nothing to do with near-term transit-traffic revenues.

More fundamentally, the real mechanism-design goal is not to convince agents to reveal their private inputs but rather to compute the desired result that depends on these inputs. The economics literature does not emphasize the fact that these are distinct goals, but the distinction a major focus of the TCS literature. The theory of *secure, multiparty function evaluation* (SMFE), developed by the cryptographic-research community, shows that functions can often be computed in such a way that nothing about agent i's private input need be revealed to agent j (except what is logically implied by the outcome and agent j's private input). In economic terms, the SMFE approach would lead to indirect mechanisms, because agents would not be revealing their utilities but instead would be using strategies drawn from some other strategy space. For an overview of SMFE, see Goldreich [12].

One cannot always apply SMFE techniques "off the shelf" to DAMD. In particular, one often cannot "compose" a direct distributed algorithmic mechanism with a standard SMFE protocol, for several fundamental reasons:

- The strategic models may be different. Some standard SMFE techniques apply to networks in which at least a constant fraction of the agents are obedient; the other agents are often assumed to be Byzantine adversaries. Although one usually does not have to design distributed

[8] The question of whether there are any such problems was brought to our attention by Eric Friedman.

mechanisms for Byzantine adversarial agents, one often has to assume that *all* of the agents will act strategically – *none* can be assumed to be obedient.

- Standard SMFE techniques for transforming an arbitrary multi-agent protocol into one that keeps agents' inputs private and computes the same output produce protocols with unacceptably high network complexity. In particular, the required total number of messages may grow quadratically (or worse) as a function of the total number of agents. Sometimes special-purpose, SMFE protocols with low network complexity are obtainable, but, if these are to be found for DAMD problems of interest, they will have to be designed on a case-by-case basis; no general SMFE results guarantee their existence.

- Some of the standard building blocks of protocols in the SMFE literature (notably secret sharing) assume that each agent knows the *set* of all agents participating in the protocol and can refer to each of them using a unique ID. Clearly this is not the case in all DAMD problems; in particular, it is not the case in the multicast cost-sharing problem, where two agents resident at different nodes of the multicast tree must be assumed to be ignorant of each other's existence.

Open Problem 5 *Explore agent privacy in specific DAMD problems of interest. More generally, devise new building blocks for SMFE protocols that are applicable in the DAMD context, where all agents can be strategic (i.e., none need be obedient or adversarial), low network complexity is crucial, and the set of participating agents is unknown to each individual agent.*

In addition to the standard SMFE literature [12], work that might be relevant to Open Problem 5 includes but is not limited to the papers of Naor and Nissim [21], Naor, Pinkas, and Sumner [22], and Dodis, Halevi, and Rabin [5].

Thus far, our motivation for considering indirect mechanisms has been the desire to preserve agents' privacy, and we have observed that low network complexity and agent privacy may be hard to achieve simultaneously. It is important to note, however, that indirect mechanisms have not been shown to have inherently higher network complexity than direct mechanisms.

Open Problem 6 *Are there DAMD problems for which all direct mechanisms have bad network complexity but at least one indirect mechanism has good network complexity?*

Indirect mechanisms might have other advantages as well, such as enabling one to trade off between agent computation and mechanism computation or to avoid worst-case running times on many instances [24]. They might also enable *approximation* of mechanism-design problems that are canonically hard.[9] These benefits have been demonstrated in the context of combinatorial auctions and should be explored in other contexts as well.

7 Acknowledgements

We thank Dirk Bergemann, Eric Friedman, Arvind Krishnamurthy, Noam Nisan, Christos Papadimitriou, David Parkes, Rahul Sami, and Richard Yang for many stimulating discussions about DAMD. We are also grateful to seminar participants at DIMACS, Stanford, Microsoft, and NEC for their comments.

[9]See Section 5 of [9] for a discussion of the meaning of approximation in AMD.

References

[1] K. Arrow, "The Property-Rights Doctrine and Demand Revelation under Incomplete Information," in **Economics and Human Welfare**, Academic Press, New York, pages 23–39, 1979.

[2] B. Bernheim, "Rationalizable Strategic Behavior," *Econometrica* **52** (1984), pages 1007–1028.

[3] E. Clarke, "Multipart pricing of public goods," *Public Choice* **11** (1971), pages 17–33.

[4] C. d'Aspremont and L.-A. Gérard-Varet, "Incentives and Incomplete Information," *Journal of Public Economics* **11** (1979), pages 25–45.

[5] Y. Dodis, S. Halevi, and T. Rabin, "A Cryptographic Solution to a Game-Theoretic Problem," in *Advances in Cryptology – Crypto 2000*, Lecture Notes in Computer Science, volume 1880, Springer, Berlin, pages 112–130, 2000.

[6] J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker, "Hardness Results for Multicast Cost Sharing," submitted; available at `http://www.cs.yale.edu/homes/jf/FKSS2.ps`. Extended abstract to appear in *Proceedings of the 2002 Conference on Foundations of Software Technology and Theoretical Computer Science.*

[7] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker, "A BGP-based Mechanism for Lowest-Cost Routing," in *Proceedings of the 21st Symposium on Principles of Distributed Computing*, ACM Press, New York, pages 173–182, 2002. Full version has been submitted for journal publication and is available at `http://www.cs.yale.edu/homes/jf/FPSS-journal.ps`.

[8] J. Feigenbaum, C. Papadimitriou, and S. Shenker, "Sharing the Cost of Multicast Transmissions," *Journal of Computer and System Sciences* **63** (2001), pages 21–41.

[9] J. Feigenbaum and S. Shenker, "Distributed Algorithmic Mechanism Design: Recent Results and Future Directions," in *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, ACM Press, New York, pages 1–13, 2002.

[10] A. Fiat, A. Goldberg, J. Hartline, and A. Karlin, "Competitive Generalized Auctions," in *Proceedings of the 34th ACM Symposium on Theory of Computing*, ACM Press, New York, pages 72–81, 2002.

[11] R. Gibbens and R. Kelly, "Resource Pricing and the Evolution of Congestion Control," *Automatica* **35** (1999), pages 1969–1985.

[12] O. Goldreich, "Secure Multi-party Computation (working draft, Version 1.1)," 1998. `http://philby.ucsd.edu/cryptolib/BOOKS/oded-sc.html`

[13] J. Green and J. Laffont. "Incentives in public decision making," in **Studies in Public Economics**, volume 1, North Holland, Amsterdam, pages 65–78, 1979.

[14] T. Griffin and G. Wilfong, "An analysis of BGP convergence properties," in *Proceedings of SIGCOMM '99*, ACM Press, New York, pages 277–288, 1999.

[15] T. Groves, "Incentives in teams," *Econometrica* **41** (1973), pages 617–663.

[16] J. Hershberger and S. Suri, "Vickrey prices and shortest paths: What is an edge worth?," in *Proceedings of the 42nd Symposium on the Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, pages 129–140, 2001.

[17] M. Jackson, "A Crash Course in Implementation Theory," *Social Choice and Welfare*, **18** (2001), pages 655–708.

[18] Y. Korilis, A. Lazar, and A. Orda, "Architecting Noncooperative Networks," *Journal on Selected Areas in Communications* **13** (1995), pages 1241–1251.

[19] E. Maskin, "Nash Equilibrium and Welfare Optimality," *Review of Economic Studies* **66** (1999), pages 23–38.

[20] H. Moulin and S. Shenker, "Strategyproof Sharing of Submodular Costs: Budget Balance Versus Efficiency," *Economic Theory* **18** (2001), pages 511–533.

[21] M. Naor and K. Nissim, "Communication-Preserving Protocols for Secure Function Evaluation," in *Proceedings of the 33rd Symposium on Theory of Computing*, ACM Press, New York, pages 590–599, 2001.

[22] M. Naor, B. Pinkas, and R. Sumner, "Privacy-Preserving Auctions and Mechanism Design," in *Proceedings of the 1st Conference on Electronic Commerce*, ACM Press, New York, pages 129–139, 1999.

[23] N. Nisan and A. Ronen, "Algorithmic mechanism design," *Games and Economic Behavior* **35** (2001), pages 166–196.

[24] D. Parkes, "Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency," PhD Thesis, University of Pennsylvania, 2001.

[25] D. Pearce, "Rationalizable Strategic Behavior and the Problem of Perfection," *Econometrica* **52** (1984), pages 1029–1050.

[26] Y. Shoham and M. Wellman, "Economic Principles of Multi-Agent Systems," *Artificial Intelligence* **94** (1997), pages 1–6.

[27] J. Smith, **Evolution and the Theory of Games**, Cambridge University Press, Cambridge, 1982.

[28] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *Journal of Finance* **16** (1961), pages 8–37.

[29] W. Walsh and M. Wellman, "A Market Protocol for Decentralized Task Allocation," in *Proceedings of the 3rd International Conference on Multi-Agent Systems*, IEEE Computer Society Press, Los Alamitos, pages 325–332, 1998.